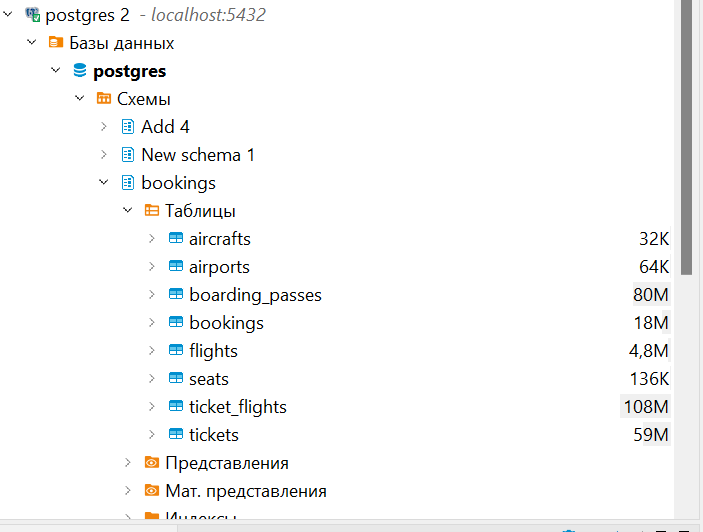
**Проектная работа по модулю**

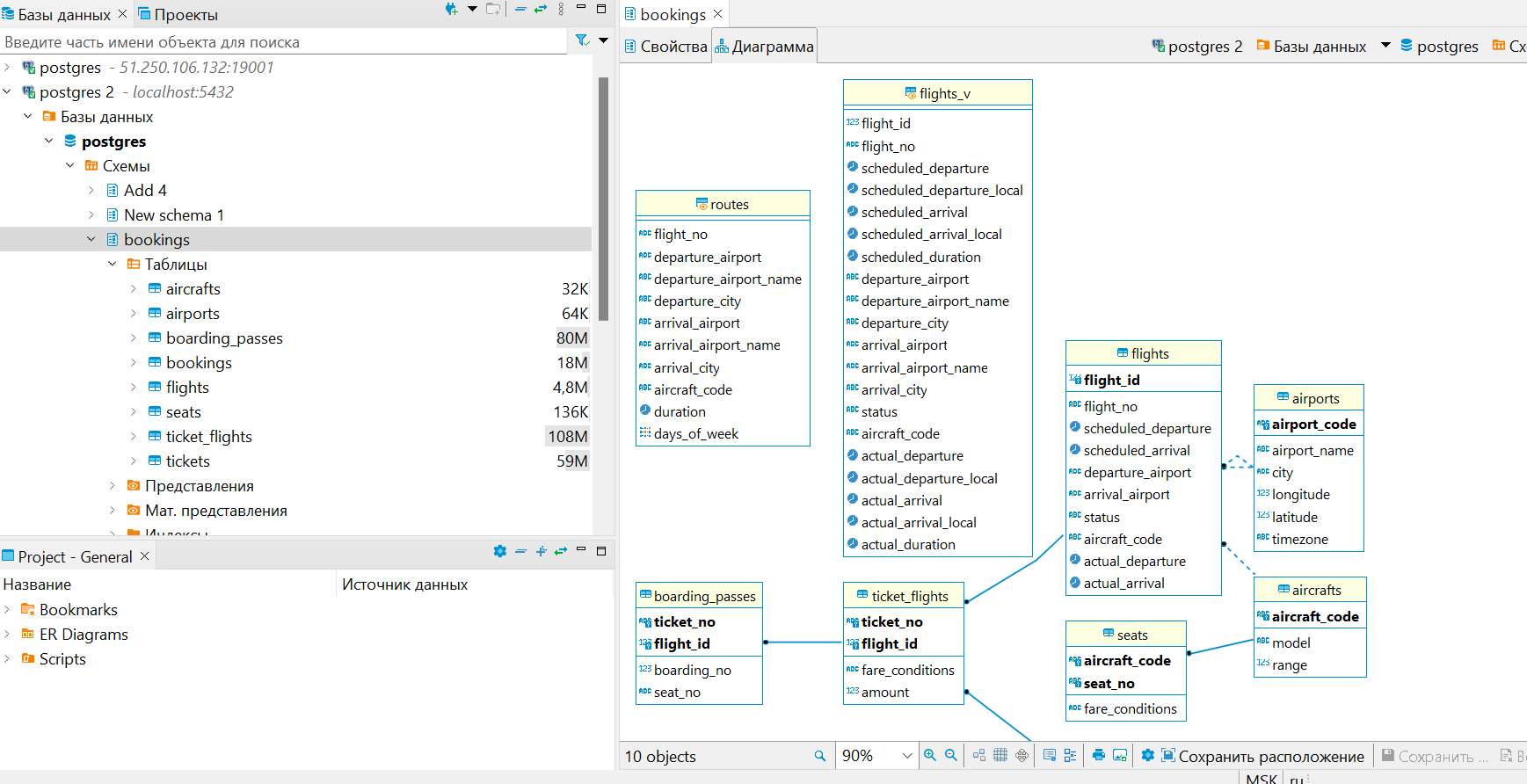
**“SQL и получение данных”**

Приложение №1

1. **В работе использовался локальный тип подключения:**

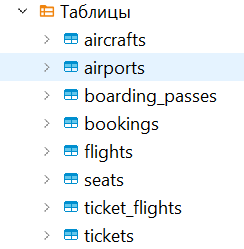


1. **Скриншот ER-диаграммы из DBeaver`a согласно Вашего подключения:**

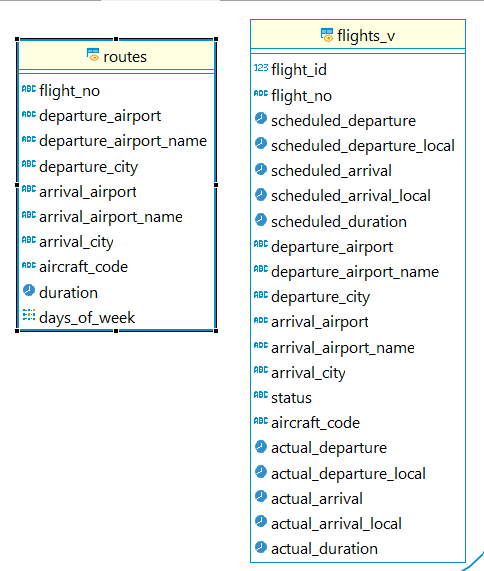


1. **Краткое описание БД - из каких таблиц и представлений состоит.**

В БД содержатся 8 таблиц (aircrafts, airports, boarding\_passes, bookings, flights, seats, ticket\_flights, tickets):



И 2 представления (routes, flights\_v):



1. **Развернутый анализ БД - описание таблиц, логики, связей и бизнес области (частично можно взять из описания базы данных, оформленной в виде анализа базы данных).**

**Описание схемы**

Основной сущностью является бронирование (bookings).

В одно бронирование можно включить несколько пассажиров, каждому из которых выписывается отдельный билет (tickets). Билет имеет уникальный номер и содержит информацию о пассажире.

Как таковой пассажир не является отдельной сущностью. Как имя, так и номер документа пассажира могут меняться с течением времени, так что невозможно однозначно найти все билеты одного человека; для простоты можно считать, что все пассажиры уникальны.

Билет включает один или несколько перелетов (ticket\_flights). Несколько перелетов могут включаться в билет в случаях, когда нет нет прямого рейса, соединяющего пункты отправления и назначения (полет с пересадками), либо когда билет взят «туда и обратно».

В схеме данных нет жесткого ограничения, но предполагается, что все билеты в одном бронировании имеют одинаковый набор перелетов.

Каждый рейс (flights) следует из одного аэропорта (airports) в другой. Рейсы с одним номером имеют одинаковые пункты вылета и назначения, но будут отличаться датой отправления.

При регистрации на рейс пассажиру выдается посадочный талон (boarding\_passes), в котором указано место в самолете. Пассажир может зарегистрироваться только на тот рейс, который есть у него в билете. Комбинация рейса и места в самолете должна быть уникальной, чтобы не допустить выдачу двух посадочных талонов на одно место.

Количество мест (seats) в самолете и их распределение по классам обслуживания зависит от модели самолета (aircrafts), выполняющего рейс. Предполагается, что каждая модель самолета имеет только одну компоновку салона. Схема данных не контролирует, что места в посадочных талонах соответствуют имеющимся в самолете (такая проверка может быть сделана с использованием табличных триггеров или в приложении).

**Объекты схемы**

Список отношений

Таблица bookings.aircrafts

Каждая модель воздушного судна идентифицируется своим трехзначным кодом (aircraft\_code). Указывается также название модели (model) и максимальная дальность полета в километрах (range).

Таблица bookings.airports

Аэропорт идентифицируется трехбуквенным кодом (airport\_code) и имеет свое имя (airport\_name). Для города не предусмотрено отдельной сущности, но название (city) указывается и может служить для того, чтобы определить аэропорты одного города. Также указывается широта (longitude), долгота (latitude) и часовой пояс (timezone).

Таблица bookings.boarding\_passes

При регистрации на рейс, которая возможна за сутки до плановой даты отправления, пассажиру выдается посадочный талон. Он идентифицируется также, как и перелет — номером билета и номером рейса. Посадочным талонам присваиваются последовательные номера (boarding\_no) в порядке регистрации пассажиров на рейс (этот номер будет уникальным только в пределах данного рейса). В посадочном талоне указывается номер места (seat\_no).

Таблица bookings.flights

Естественный ключ таблицы рейсов состоит из двух полей — номера рейса (flight\_no) и даты отправления (scheduled\_departure). Чтобы сделать внешние ключи на эту таблицу компактнее, в качестве первичного используется суррогатный ключ (flight\_id). Рейс всегда соединяет две точки — аэропорты вылета (departure\_airport) и прибытия (arrival\_airport). Такое понятие, как «рейс с пересадками» отсутствует: если из одного аэропорта до другого нет прямого рейса, в билет просто включаются несколько необходимых рейсов. У каждого рейса есть запланированные дата и время вылета (scheduled\_departure) и прибытия (scheduled\_arrival). Реальные время вылета (actual\_departure) и прибытия (actual\_arrival) могут отличаться: обычно не сильно, но иногда и на несколько часов, если рейс задержан. Статус рейса (status) может принимать одно из следующих значений:

• Scheduled Рейс доступен для бронирования. Это происходит за месяц до плановой даты вылета; до этого запись о рейсе не существует в базе данных.

• On Time Рейс доступен для регистрации (за сутки до плановой даты вылета) и не задержан.

• Delayed Рейс доступен для регистрации (за сутки до плановой даты вылета), но задержан.

• Departed Самолет уже вылетел и находится в воздухе.

• Arrived Самолет прибыл в пункт назначения.

• Cancelled Рейс отменен.

Таблица bookings.seats

Места определяют схему салона каждой модели. Каждое место определяется своим номером (seat\_no) и имеет закрепленный за ним класс обслуживания (fare\_conditions) — Economy, Comfort или Business.

Таблица bookings.ticket\_flights

Перелет соединяет билет с рейсом и идентифицируется их номерами. Для каждого перелета указываются его стоимость (amount) и класс обслуживания (fare\_conditions).

Таблица bookings.tickets

Билет имеет уникальный номер (ticket\_no), состоящий из 13 цифр. Билет содержит идентификатор пассажира (passenger\_id) — номер документа, удостоверяющего личность, — его фамилию и имя (passenger\_name) и контактную информацию (contact\_date). Ни идентификатор пассажира, ни имя не являются постоянными (можно поменять паспорт, можно сменить фамилию), поэтому однозначно найти все билеты одного и того же пассажира невозможно.

Представление "bookings.flights\_v"

Над таблицей flights создано представление flights\_v, содержащее дополнительную информацию:

• расшифровку данных об аэропорте вылета (departure\_airport, departure\_airport\_name, departure\_city),

• расшифровку данных об аэропорте прибытия (arrival\_airport, arrival\_airport\_name, arrival\_city),

• местное время вылета (scheduled\_departure\_local, actual\_departure\_local),

• местное время прибытия (scheduled\_arrival\_local, actual\_arrival\_local),

• продолжительность полета (scheduled\_duration, actual\_duration).

Материализованное представление bookings.routes

Таблица рейсов содержит избыточность: из нее можно было бы выделить информацию о маршруте (номер рейса, аэропорты отправления и назначения), которая не зависит от конкретных дат рейсов. Именно такая информация и составляет материализованное представление routes.

**Бизнес задачи, которые можно решить, используя БД.**

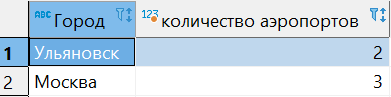
1. Получить статистику по рейсам – в каких аэропортах чаще всего задерживаются рейсы, в каких чаще всего улетают вовремя, и есть ли связь с типом самолета.
2. Получить статистику по классам путешествий – есть ли зависимость класса путешествия от дальности полета.
3. Получить данные по количеству непрошедших регистрацию пассажиров, а также в каких аэропортах чаще всего пассажиры не проходят регистрацию.
4. Получить статистику по загруженности каждого аэропорта – количество вылетевших рейсов и пассажиров на них.
5. Получить информацию по самым популярным направлениям по городам – частота рейсов и количество пассажиров.
6. Получить информацию по средней стоимости перелета, зависит ли стоимость от дальности полета.
7. **Список SQL запросов из приложения №2 с описанием логики их выполнения.**
8. В каких городах больше одного аэропорта?

**select** city **as** "Город", **count**(airport\_code) **as** "количество аэропортов" --- перечисляю столбцы с информацией по городу и количеству аэропортов в этом городе

**from** airports a --- из таблицы в аэропортами

**group** **by** city --- группирую по городу (чтобы по одному городу была одна строка)

**having** **count**(airport\_code) > 1 --- добавлю фильтр, что в результате нужны только те города, где количество аэропортов больше 1



1. В каких аэропортах есть рейсы, выполняемые самолетом с максимальной дальностью перелета? (Подзапрос)

**select** **distinct** f.departure\_airport "Аэропорт вылета", a.city **as** "Город вылета", t.model "Модель самолета", **count**(t.aircraft\_code) **as** "Количество самолетов", t."range" "Дальность полета км" --- перечисляю список столбцов, которые нужно показать в итоговом результате (дополнительно добавляю названия городов, модель самолета и количества самолетов в аэропортах для наглядности – не обязательные столбцы) и задаю алиасы

**from** (**select** aircraft\_code, model, "range" --- в подзапросе нахожу самолет с дальностью его полета

**from** aircrafts --- из таблицы самолеты

**order** **by** "range" **desc** --- сортирую по убыванию дальности полета

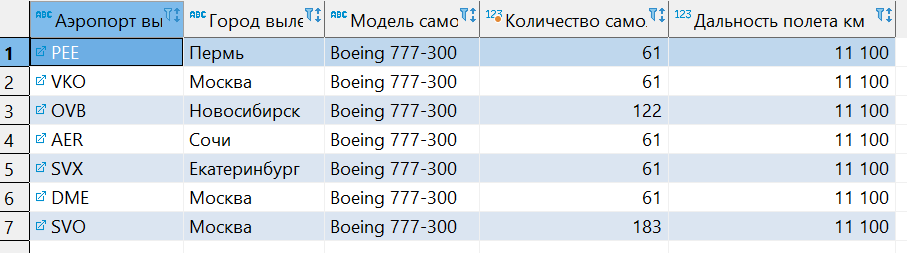
**limit** 1 --- задаю ограничение 1 (с максимальной дальностью)

) t

**join** flights f **on** f.aircraft\_code = t.aircraft\_code --- соединяю с таблицей перелеты по уникальному коду самолета

**join** airports a **on** a.airport\_code = f.departure\_airport --- соединяю с таблицей аэропорты по аэропорту вылета (не обязательно, просто я хотела вывести информацию по городам, т.к. код аэропорта не такой информативный)

**group** **by** 1, 2, 3, 5 --- группирую по 1, 2, 3, 5 столбцам



1. Вывести 10 рейсов с максимальным временем задержки вылета (Оператор LIMIT)

**select** a.city **as** "Город вылета", a2.city "Город назначения", f.flight\_no **as** "Номер рейса", (f.actual\_departure - f.scheduled\_departure) **as** "Задержка вылета" --- перечисляю список столбцов, которые нужно показать в итоговом результате (дополнительно добавляю названия городов для наглядности – не обязательные столбцы) и задаю алиасы

**from** flights f --- из таблицы перелеты

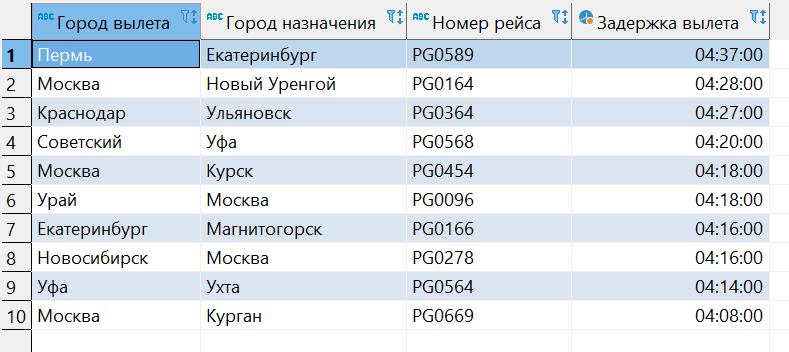
**join** airports a **on** a.airport\_code = f.departure\_airport --- соединяю с таблицей аэропорты по аэропорту вылета (не обязательно, просто я хотела вывести информацию по городам, т.к. номер рейса не такой информативный)

**join** airports a2 **on** a2.airport\_code = f.arrival\_airport --- соединяю с таблицей аэропорты по аэропорту прилета (не обязательно, просто я хотела вывести информацию по городам, т.к. номер рейса не такой информативный)

**where** (f.actual\_departure - f.scheduled\_departure) **is** **not** **null** --- добавляю условие, что в результат должны попасть ненулевые значения результата вычитания фактического времени вылета и запланированного времени вылета

**order** **by** 4 **desc** --- упорядочиваю по столбцу с задержкой рейса по убыванию

**limit** 10 --- вывожу 10 результатов с максимальным временем задержки вылета



1. Были ли брони, по которым не были получены посадочные талоны? (Верный тип JOIN)

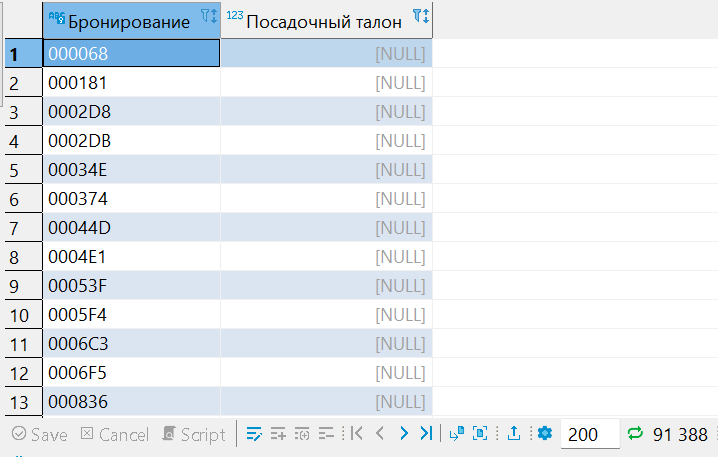
**select** t.book\_ref "Номер бронирования", bp.boarding\_no "Номер посадочного талона" --- перечисляю список столбцов, которые нужно показать в конечном результате

**from** tickets t --- из таблицы с билетами

**left** **join** boarding\_passes bp **on** bp.ticket\_no = t.ticket\_no --- присоединяю таблицу с посадочными талонами по уникальному номеру билета

**where** bp.boarding\_no **is** **null** --- задаю условие, чтобы вывелись только нулевые значения для номера посадочного талона

**group** **by** 1, 2 --- группирую данные



1. Найдите количество свободных мест для каждого рейса, их % отношение к общему количеству мест в самолете. *---- сделана только первая часть*

Добавьте столбец с накопительным итогом - суммарное накопление количества вывезенных пассажиров из каждого аэропорта на каждый день. Т.е. в этом столбце должна отражаться накопительная сумма - сколько человек уже вылетело из данного аэропорта на этом или более ранних рейсах в течении дня. (Оконная функция, Подзапросы или/и cte)

**with** b **as** --- в CTE нахожу количество севших на борт пассажиров

(**select**

f.flight\_id,

**count**(bp.boarding\_no) **as** b\_qty

**from** flights f

**join** boarding\_passes bp **on** bp.flight\_id = f.flight\_id

**where** f.actual\_departure **is** **not** **null**

**group** **by** f.flight\_id),

ss **as** --- в CTE нахожу количество мест в самолете

(**select**

a.aircraft\_code,

**count**(s.seat\_no) **as** seat\_qty

**from** seats s

**join** aircrafts a **on** a.aircraft\_code = s.aircraft\_code

**group** **by** a.aircraft\_code)

**select** f.flight\_id **as** "Идентификатор рейса", --- перечисляю список столбцов, которые нужно показать в конечном результате

f.flight\_no **as** "Номер рейса",

f.departure\_airport **as** "Аэропорт вылета",

f.actual\_departure::**date** **as** "Время вылета по факту",

b.b\_qty **as** "Кол-во человек на борту",

ss.seat\_qty **as** "Кол-во мест в самолете",

**sum**(ss.seat\_qty - b.b\_qty) **as** "Кол-во свободных мест в самолете",

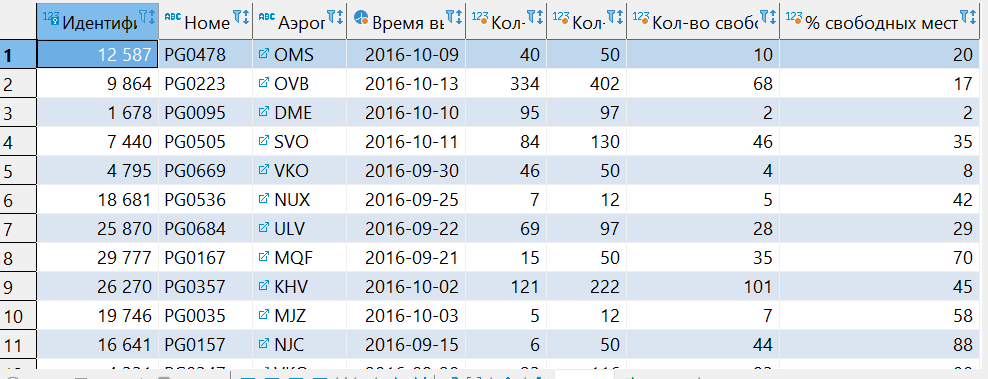
**round**(**sum**(ss.seat\_qty - b.b\_qty)::**numeric** / ss.seat\_qty::**numeric**, 2) \* 100 **as** "% свободных мест от общего"

**from** flights f

**join** b **on** b.flight\_id = f.flight\_id --- присоединяю таблицу CTE

**join** ss **on** ss.aircraft\_code = f.aircraft\_code --- присоединяю таблицу CTE

**group** **by** 1, 5, 6 --- группирую



1. Найдите процентное соотношение перелетов по типам самолетов от общего количества. (Подзапрос или окно, Оператор ROUND)

**select** model **as** "Тип самолета", qty **as** "Кол-во перелетов", (**round**(qty / (**sum**(qty) **over** ()), 2) \* 100) **as** "% от общего кол-ва перелётов" --- перечисляю список столбцов, которые нужно показать в конечном результате, также добавляю столбей с расчетом % от общего кол-ва перелетов для каждого самолета (столбце с кол-вом перелетов необязятельный)

**from**(

**select** **count**(flight\_id) **as** qty, model --- в подзапросе нахожу количество перелетов для каждого самолета

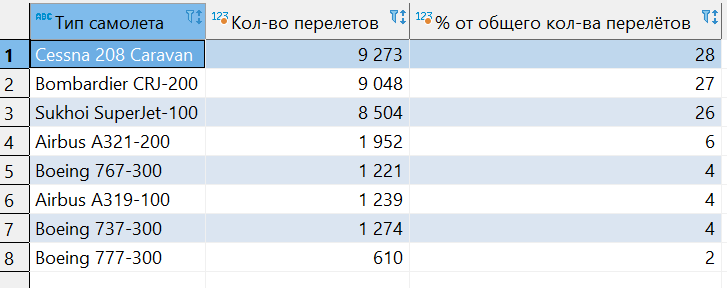
**from** flights f

**join** aircrafts a **on** a.aircraft\_code = f.aircraft\_code --- для этого присоединяю таблицу с самолетами

**group** **by** model --- группирую по модели

) l

**order** by 3 desc --- упорядочиваю по 3 столбцу (не обязательно)



1. Были ли города, в которые можно добраться бизнес - классом дешевле, чем эконом-классом в рамках перелета? (CTE)

**with** prices **as**( --- в CTE находим тарифы по бизнес-классу

**select**

a.city "Город отправления",

a2.city "Город прибытия",

tf.fare\_conditions,

**case** **when** tf.fare\_conditions = 'Business' **then** **min**(tf.amount) **end** b\_min\_amount, --- минимальные значения за перелет

**case** **when** tf.fare\_conditions = 'Economy' **then** **max**(tf.amount) **end** e\_max\_amount --- максимлаьные значения за перелет

**from** flights f

**join** ticket\_flights tf **on** tf.flight\_id = f.flight\_id --- присодиняю таблицу с информацией по билетам-перелетам по уникальному идентификатору рейса

**join** airports a **on** f.departure\_airport = a.airport\_code --- присоединяю таблицу аэропорты по аэропорту отправления

**join** airports a2 **on** f.arrival\_airport = a2.airport\_code --- присоединяю таблицу аэропорты по аэропорту прибытия

**group** **by** 1, 2, 3 --- группирую

)

**select** --- перечисляю список столбцов, которые нужно показать в конечном результате

"Город отправления",

"Город прибытия",

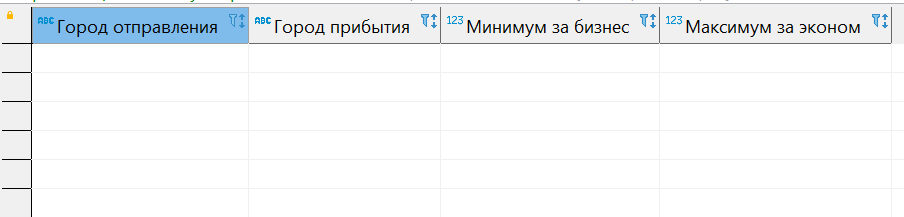
**min**(b\_min\_amount) "Минимум за бизнес",

**max**(e\_max\_amount) "Максимум за эконом"

**from** prices --- из созданной CTE

**group** **by** 1, 2 --- группирую по городам

**having** **min**(b\_min\_amount) < **max**(e\_max\_amount) --- добавляю фильтр, что нужно показать только те города, по которым максимальная цена за перелет эконом классом больше, чем минимальная цена за перелет бизнес-классом



1. Между какими городами нет прямых рейсов? (Декартово произведение в предложении FROM, Самостоятельно созданные представления (если облачное подключение, то без представления), Оператор EXCEPT)

**create** **view** cities\_v **as** --- Создаю представление city\_v с городами орпавления-прибытия

**select** v.departure\_city, v.arrival\_city

**from** flights\_v v

**select** **distinct** a.city, a1.city --- с помощью distinct убираю дубли (в Москве и Ульяновске больше 1 аэропорта)

**from** airports a

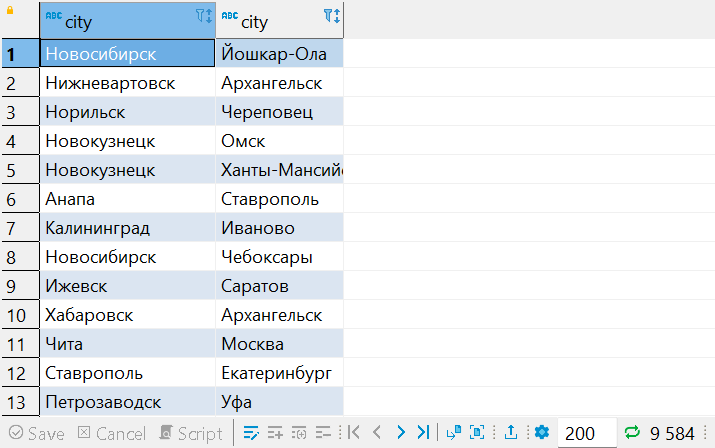
**cross** **join** airports a1 --- используя декартовое произведение нахожу связи город-город

**where** a.city != a1.city --- убираю связки городов на себя

**except** --- из результата убираю данные, которые есть в представлении

**select** c.departure\_city, c.arrival\_city

**from** cities\_v c



1. Вычислите расстояние между аэропортами, связанными прямыми рейсами, сравните с допустимой максимальной дальностью перелетов в самолетах, обслуживающих эти рейсы \* (Оператор RADIANS или использование sind/cosd, CASE)

**select** **distinct** a.city **as** "Город отправления", a2.city **as** "Город прибытия",

a3.model **as** "Тип самолета", a3."range" **as** "Макс дальность полета самолета", --- перечисляю столбцы, которые надо итогово вывести

**round**(((**acos**((**sind**(a.latitude)\***sind**(a2.latitude) + **cosd**(a.latitude) \* **cosd**(a2.latitude) \* **cosd**((a.longitude - a2.longitude))))) \* 6371)::**numeric**, 2)

**as** "Расстояние между городами", --- использую сложную формулу из задания для вычисления расстояний между городами по координатам, умножение на 6371 необходимо, чтобы получить километры

**case** **when** --- задаю условие

a3."range" >= **round**(((**acos**((**sind**(a.latitude)\***sind**(a2.latitude) + **cosd**(a.latitude) \* **cosd**(a2.latitude) \* **cosd**((a.longitude - a2.longitude))))) \* 6371)::**numeric**, 2) --- если вычисленное расстояние между городами больше, чем заданная дальность полета самолета

**then** 'OK' --- тогда в столбец со сравнением вывести ОК

**else** 'not possible' --- если вычисленное расстояние больше возможной дальности полета самолета, то вывести not possible

**end** "Сравнение" --- название столбца, куда выводится результат ОК \ not possible

**from** flights f

**join** airports a **on** a.airport\_code = f.departure\_airport --- присоединение таблицы с аэропортами

**join** airports a2 **on** a2.airport\_code = f.arrival\_airport --- присоединение таблицы с аэропортами

**join** aircrafts a3 **on** a3.aircraft\_code = f.aircraft\_code --- присоединение таблицы с самолетами

**order** **by** 5 **desc** --- упорядочиваю по расстоянию между городами

